FINAL PROJECT FOR CSCI5229

XU HAN

CONTENTS

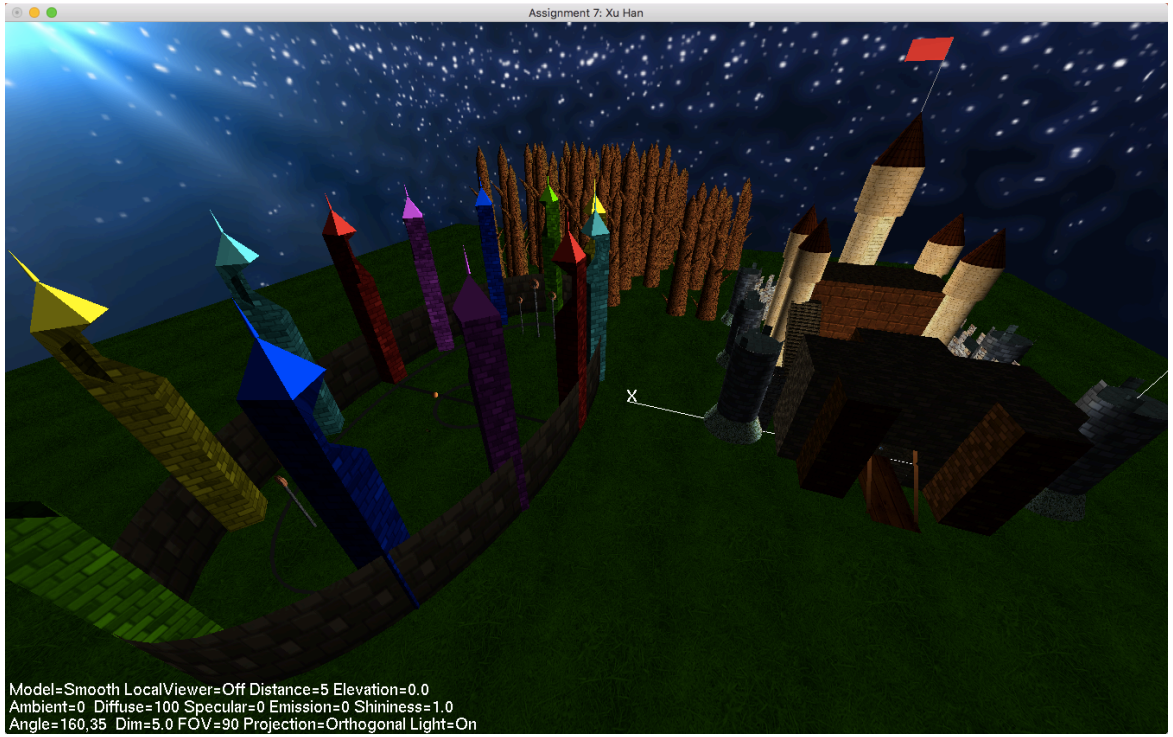
- Basic description
- Demonstration
- Implementation

CONTENTS

- **Basic description**
- Demonstration
- Implementation

BASIC DESCRIPTION

- A simple implementation of Harry Potter’ s world
- It is a little game that allows users to experience this magic world and interact with the scene.



BASIC DESCRIPTION --CASTLE

- In this part, we will find that there is fire simulation inside the Lounge. We could rest here and do some little magic tricks with the teapot on the table, like making it move with our wand or enlarging it.

- Amazing parts:
- Fire simulation and light modeling (particle engine)
- 3D selection

BASIC DESCRIPTION --FOREST

- In this part, there is a pool in the deep forest. Beside the pool, we could use our wand to call a patronum and light the forest.

- Amazing parts:
- Water wave simulation
- Light modeling with patronum and wand light

BASIC DESCRIPTION --COURT

- In this part, we could use our wand to set off fireworks and play Quidditch. When playing Quidditch, we need to first select the golden snitch, the ball, the flying broomstick or the bat. The broomstick will help us fly and we will either chase the golden snitch or use the bat to hit the ball.

- Amazing parts:
- 3D selection
- Ball runway calculation
- Firework simulation (particle engine)

CONTENTS

- Basic description
- **Demonstration**
- Implementation

DEMONSTRATION

CONTENTS

- Basic description
- Demonstration
- **Implementation**

IMPLEMENTATION --PARTICLE ENGINE

- Two class definition, one is for particle, one is for particle system
- Particle class: position, color, velocity, pointer to particle system class, etc. Initialize(), update(), render()
- Particle system class: position, emit direction, number of particles, age, pointer to particle class, etc. Initialize(), updateSystem(), render(), ect.
- Particle.render(): GL_POINTS for firework and GL_POLYGON for fire simulation

IMPLEMENTATION --WATER WAVE SIMULATION

- Define a struct and a vector. The vector is used to store all these structs to create a pool.
- I randomly choose two structs in different position as exciters. They change their Y position following sin().
- All the structs' s Y position will be changed following their four surrounding structs. And their normal vectors are calculated by doing cross product.
- Use glDrawElements() to draw the pool. Use glVertexPointer(), glNormalPointer() to pass the pointers.

```
struct SOscillator
{
  GLfloat x,y,z;
  GLfloat nx,ny,nz;  //normal vector
  GLfloat UpSpeed;
  GLfloat newY;
  bool bIsExciter;
  //only in use, if bIsExciter is true:
  float ExciterAmplitude;
  float ExciterFrequency;
};
```

IMPLEMENTATION --3D SELECTION

- Render the objects to be selected: the broomstick, the ball, the bat, the golden snitch and the teapot
- Retrieve the objects' ID according to the mouse's position.
- Use glGetIntegerv() to get the current viewport coordinates; use gluPickMatrix() and gluPerspective() to generate the current matrix of the viewport; use glRenderMode() to select the object; use glPushName() to build name stack.

```
int RetrieveObjectID(int x, int y)
{
  int objectsFound = 0;               // This will hold the amount of objects clicked
  int viewportCoords[4] = {0};

  unsigned int selectBuffer[32] = {0};
  glSelectBuffer(32, selectBuffer);         // Setup selection buffer to accept object ID's
  glGetIntegerv(GL_VIEWPORT, viewportCoords);     // Get the current view port coordinates
  glMatrixMode(GL_PROJECTION);            // effect projection matrix

  glPushMatrix();
   glRenderMode(GL_SELECT);
    glLoadIdentity();               // Reset projection matrix
    gluPickMatrix(x, viewportCoords[3] - y, 20, 20, viewportCoords);
    gluPerspective(45.0f,(float)width_win/(float)height_win,0.1f,150.0f);
    glMatrixMode(GL_MODELVIEW);           // Go back into model view matrix
    RenderScene();
    objectsFound = glRenderMode(GL_RENDER);
    glMatrixMode(GL_PROJECTION);
  glPopMatrix();

  glMatrixMode(GL_MODELVIEW);

  if (objectsFound > 0)
  {
    int selectedObject = selectBuffer[3];
    return selectedObject;
  }
  return 0;
}
```

IMPLEMENTATION --RUN WAY CALCULATION

- Find the hitting position and calculate a sphere’ s function. The ball will run through the line, which is decided by the current hitting point and the point on the sphere.

- When changing the view angle and the bat’ s angle, the point on the sphere will also be changed.

```
float Position_ball[]  = {float(8*t_bat*2*Cos(zh_bat)/(8*Sin(zh_bat)*Cos(zh)+3.0)-3.0),
                          float(8*t_bat*2*Cos(zh_bat)/(8*Sin(zh_bat)*Sin(zh)-4.0)+4.0),
                          float(t_bat*2)};
```

REFERENCE

- Angel, Edward; OpenGL: A Primer; 3rd Edition
- https://github.com/r0bbE/ParticleEngine
- http://www.360doc.com/content/14/1028/10/19175681_420522404.shtml

THANKS